Fast roster generation for single-nurse rostering problems

Robin Tourlamain[1,2][0009−0003−3782−8063], Pieter Smet[2][0000−0002−3955−7725], and Greet Vanden Berghe[2][0000−0002−0275−5568]

[1] The University of Melbourne, School of Mathematics and Statistics
Parkville, VIC 3010, Australia
[2] KU Leuven, Department of Computer Science, NUMA
Gebroeders de Smetstraat 1, 9000 Gent, Belgium
{robin.tourlamain, greet.vanden.berghe, pieter.smet}@kuleuven.be

Abstract. Despite significant research into nurse rostering problems, generating feasible solutions for large instances that include complex hard constraints as part of nurses' contracts, and proving their optimality, remains a challenge. The combination of long rostering periods and many shift types leads to enormous search trees that simply cannot be fathomed by current exact algorithms. In this paper we help close this gap by developing effective pruning strategies for dynamic programming approaches, drastically reducing the number of individual rosters that must be considered, as well as a heuristic that quickly identifies feasible rosters. We demonstrate how incorporating these strategies into state-of-the-art methods significantly speeds up the computation of optimal (partial) solutions. The strategies we suggest for rostering a single nurse in isolation can be embedded within a complete nurse rostering approach.

Keywords: Nurse rostering · Dynamic programming · Pruning strategies

1 Introduction

Nurse Rostering Problems (NRPs) require the assignment of shifts to nurses throughout a given rostering period so that shift staffing demand is met while respecting various constraints that arise from the nurses' contracts. Numerous variants of these problems have been studied, differing in terms of their objectives and the extent to which certain constraints are relaxed. This paper focuses on a specific NRP first introduced by Curtois et al. [6], which distinguishes itself from other NRPs by featuring large rostering periods of up to one year and many hard constraints at the single-nurse level. This complicates the generation of optimal or simply just feasible rosters for each individual nurse, a problem we will refer to as the single-nurse rostering problem (SNRP).

Exact methods for this specific NRP rely on decomposing the problem into multiple SNRPs [4,12], where each SNRP is solved using Dynamic Programming (DP). However, even the best algorithms currently available in the academic literature are unable to obtain optimal solutions for large instances of this problem.

We propose an extension of the DP algorithm for the SNRP described by Legrain et al. [12] that enables stronger pruning of the roster solution space, thereby accelerating the search for optimal rosters. Additionally, we introduce a guiding heuristic that efficiently generates feasible rosters for the largest SNRPs while preserving the guarantee of global optimality.

The remainder of this paper is structured as follows. First, we review the related literature on SNRPs in Section 2. Next, Section 3 presents a definition of the specific NRP we will address along with a brief review of the model for which we will develop our algorithmic improvements. In Section 4 we introduce various strategies for accelerating optimal roster generation, while Section 5 focuses on finding feasible solutions quickly. A computational study in which we quantify the impact of our contributions is presented in Section 6. Finally, Section 7 concludes the paper.

2 Related work

Although the SNRP has received limited attention as a standalone problem, it is often studied when decomposing a NRP. For example, Aickelin et al. [1] first use a genetic algorithm to construct a set of feasible rosters for each nurse, before assembling a combination into the full roster. Although many studies adopt this two-phase approach, the methods used to solve the SNRPs differ considerably. For example, Demassey et al. [7] generate single-nurse rosters using constraint programming, while Zhang et al. [18] formulate the SNRP as an integer programming network flow model. The winners of the second International Nurse Rostering Competition [5] also developed a network flow formulation in which the nurses' contractual constraints were modeled as part of the network structure [15]. The SNRP can itself be further decomposed, as demonstrated by Brucker et al. [3], who generate short feasible shift sequences and then combine these into a single-nurse roster using greedy heuristics. Legrain et al. [13] apply a similar approach and construct the final roster using a network flow integer programming formulation.

One approach to generate single-nurse rosters that is effective in practice is to reformulate the SNRP as a Shortest Path Problem with Resource Constraints (SPPRC), where each complete path corresponds to a feasible roster for a single nurse. Jaumard et al. [11] first generate all feasible paths and apply a standard shortest path algorithm to find an optimal solution. Maenhout et al. [14] improve upon this method by introducing dominance rules, eliminating the need to explicitly enumerate all feasible paths. Burke et al. [4] further reduce computation time through a beam search strategy, resorting to a full tree search only if no desirable roster is found. Strandmark et al. [16] show that solving the SPPRC to optimality is not necessary to produce high-quality rosters using their heuristic method. Dohn et al. [9] decompose the SNRP and solve the SPPRC over a set of work stretches instead of individual shifts.

The state-of-the-art approach for NRPs was developed by Legrain et al. [12], who have the best results for multiple nurse rostering datasets using their

SPPRC-based branch-and-price method. They were able to achieve this by introducing *strong* dominance rules in their DP algorithm for the SNRP, which not only consider the current state of a partial roster, but also the remaining possible completions. However, while they employ various acceleration techniques such as bidirectional search and beam search, solving the largest SNRPs remains a computational challenge.

3 Problem definition

This section begins by describing the considered NRP in Section 3.1, before moving on to the SPPRC model of the SNRP which forms the basis of the current best algorithm [12] in Section 3.2.

3.1 Nurse rostering

Curtois et al. [6] introduce a NRP where nurses are limited to at most one shift per day. The roster is subject to a variety of constraints which are categorized as either hard (violations forbidden) or soft (violations penalized), with the objective being to minimize the total sum of penalties.

For each day and shift type, soft coverage constraints indicate the minimum number of worked shifts required across all nurses as well as a maximum number to avoid over-staffing.

Furthermore, each nurse has a personalized contract that enforces a set of individual constraints which are especially relevant for the SNRP. Bilgin et al. [2] identify three types of nurse-specific constraints and group them based on the purpose they serve in the roster. Two of these constraint types appear in the NRP from Curtois et al. [6]: *counter* and *series* constraints. Each nurse's contract contains hard counter constraints, limiting the total number of hours or weekends a nurse may work. A contract may also specify individual counter constraints for each shift type. Additionally, each nurse is subject to series constraints which dictate a minimum and maximum number of consecutive days they may work. A second series constraint imposes a minimum number of consecutive days idle, with no upper limit. Each nurse is also assigned a unique set of days on which they cannot work any shift and must remain idle.

All nurses must respect hard shift succession constraints that forbid them to work certain shifts immediately after another. Finally, a nurse may indicate preferences to work or not work certain shifts on certain days. Satisfying these preferences is considered part of the objective.

3.2 SPPRC model

The improvements we will propose in Section 4 are based on a common SPPRC formulation of the SNRP [4,16,12] that Legrain et al. [12] described clearly. Therefore, in this section we will restrict ourselves to providing a brief overview of relevant notation and the most important concepts of the SPPRC model.

The single-roster search space is represented by a directed acyclic graph $G = (V, A)$, with a set of vertices V and set of arcs A. Each vertex v represents a unique day-shift combination, with arcs corresponding to the valid transitions between them. Arcs flow from an artificial source o to an artificial sink t. Graph G in Figure 1 provides an example that features a period of four days (d) and two shift types (s), where vertices at equal distance from o represent the distinct shift types available on that day. For each day, a dummy vertex s_0 is introduced to represent the absence of a worked shift. Constraints are modeled as bounded resources that are consumed when traversing certain arcs, with R denoting the set of all resources. Let P be a partial path from o to v. For any partial path P in G, the consumption of such a resource r is denoted $\gamma^r(P)$. For example, to enforce a constraint on the maximum number of hours worked, a bounded resource r_w is defined that must satisfy $\gamma^{r_w}(P) \leq U^{r_w}$, where U^{r_w} represents the maximum number of hours worked. Upon entering a vertex v, the number of hours in the corresponding shift type is added to $\gamma^{r_w}(P)$. A complete and feasible roster is any path from o to t that satisfies all resource constraints. The cost of a path P, denoted $\gamma^0(P)$, is defined by the sum of all penalties incurred by violations of soft constraints. A completion of P is a path from o to t that includes P itself and is denoted as $[P, \overline{P}]$, where $\overline{P}$ is the path from v to t. We refer to $\overline{P}$ as the suffix.

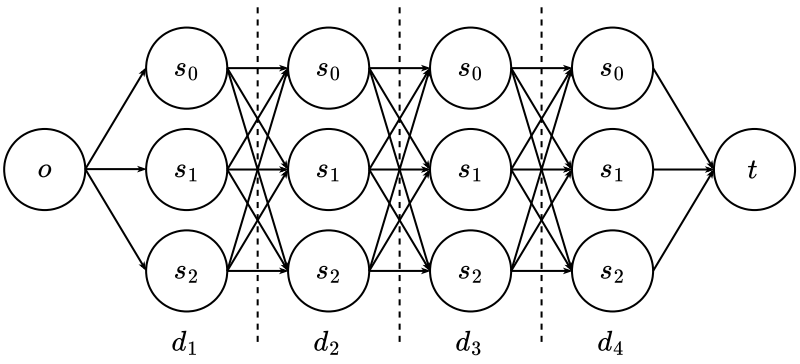

Fig. 1. Example of the graph G for a SNRP instance with a rostering period of four days and two shift types.

Building a solution involves iteratively extending partial paths along the arcs, starting from source o. With every extension, the resources consumed along the path are updated using Resource Extension Functions (REFs), and the cost is adjusted based on Cost Extension Functions (CEFs). Desaulniers et al. [8] employed this terminology in the context of vehicle routing, with Legrain et al. [12] later formalizing it for the NRP. We refer to Legrain et al. [12] for detailed definitions of the REFs and CEFs of the SNRP we are addressing in this paper. The feasible path from o to t with the minimum cost corresponds to an optimal single-nurse roster.

4 Partial path dominance

In practice, extending all feasible paths through graph G is intractable for large instances due to computation time or hardware limitations. Fortunately, we can prove that some partial paths are dominated and can thus be safely pruned without ever eliminating the optimal solution. The general dominance criterion is defined by Legrain et al. [12] as follows:

Dominance: *Let P and Q be two distinct paths from origin o to a vertex v. Path P dominates Q if and only if any feasible completion of Q, $[Q, \overline{Q}]$, yields a feasible completion of P, $[P, \overline{Q}]$, such that $\gamma^0\left([P, \overline{Q}]\right) \leq \gamma^0\left([Q, \overline{Q}]\right)$, and if there is a feasible completion of Q, $[Q, \tilde{Q}]$ such that $\gamma^0\left([P, \tilde{Q}]\right) < \gamma^0\left([Q, \tilde{Q}]\right)$.*

For a path P to dominate another path Q, their resources must satisfy a set of *dominance rules*. Let L^r and U^r denote the lower and upper bounds on resource $r \in R$. For example, for a series constraint on consecutive days worked, L^r and U^r correspond to the minimum and maximum allowed number of feasible consecutive assignments. Let $\overline{D}$ be the number of days remaining in the rostering period after vertex v, and let us define $[x]^+ = \max(x, 0)$. The dominance rules state that P dominates Q with respect to resource r if:

$$\left[\gamma^r(P) + \overline{D} - U^r\right]^+ \leq \left[\gamma^r(Q) + \overline{D} - U^r\right]^+ \text{ and } [L^r - \gamma^r(P)]^+ \leq [L^r - \gamma^r(Q)]^+ \quad (1)$$

Legrain et al. [12] conducted extensive computational experiments with various acceleration strategies for DP and found that bidirectional path extension is the most effective for reducing computation time. In the following sections, we will extend their bidirectional dominance-based path extension algorithm to improve its performance.

4.1 Inter-shift dominance (ISD)

Two partial rosters can only be compared using dominance rules (1) if they both terminate at the same vertex, meaning they must end with the same shift type. This rule originates from vehicle routing problems [10], where vertices represent locations. However for the SNRP, this needlessly excludes other valid comparisons. We will therefore introduce a more precise rule to facilitate *inter-shift dominance*.

Recall that for P to dominate Q, all feasible completions of Q must also be feasible completions of P. The identical vertex criterion simply avoids that $\overline{Q}$ is an infeasible completion of P due to a forbidden shift succession between the last shift of P and the first shift of $\overline{Q}$. However, we can guarantee this property at the shift type level. Indeed, for this property to hold it suffices that the set of feasible shift successions for P includes at least those that are feasible for Q.

Let $j \in V$ be a vertex corresponding to a certain shift type. The set of neighbouring vertices corresponds to the set of feasible shift successions and is defined as $N^+(j) = \{k \in V \mid (j, k) \in A\}$. Let P be a partial path from origin o

to u, while Q is a partial path from o to v. P dominates Q with respect to the succession constraints if:

$$N^+(v) \subseteq N^+(u)$$

4.2 Maximum remaining resource consumption (MRC)

The efficacy of dominance rules (1) can be improved by substituting $\overline{D}$ with a better approximation of the resource's maximum remaining consumption (MRC). As $\overline{D}$ gets smaller, more paths become equivalent with respect to r, allowing a larger number of paths to be pruned away. In what follows, we will define improved approximations for several constraint-associated resources by incorporating information from the relevant constraint parameters. Note that the approximated MRC at a vertex v must not underestimate the true MRC, which ensures that paths which could lead to a better complete roster will not be pruned.

Maximum number of consecutive days worked The MRC of this constraint's resource is equal to $\overline{D}$ under the assumption that the nurse in question is available for work during the remainder of the period. However, many rosters include fixed days off, forcing the stretch of days worked to end prematurely. Thus, we replace $\overline{D}$ with the number of days remaining until the next fixed day off.

Workload constraints For constraints concerning the number of days worked and the number of assignments of a specific shift type, the MRC is closely related to the maximum number of days available for work in the remainder of the rostering period. We can therefore approximate the MRC of this resource.

The presence of hard constraints on the maximum number of consecutive days worked implies that, in most cases, the actual maximum number of days worked in any suffix is less than $\overline{D}$. Consider a pattern with length $B = U^{\text{work}} + L^{\text{off}}$ defined by working the maximum number of consecutive days U^{work} followed by the minimum number of consecutive days off L^{off}. Repeating this pattern for the remaining rostering period provides a good approximation of resource r's MRC.

For an arbitrary number of remaining days $\overline{D}$, we first compute the number of full patterns f that fit within this period, the number of trailing days t after working those patterns, and the subset of those days t_w available for work. The approximation of the MRC of working days MRC^r is:

$$f = \left\lfloor \frac{\overline{D}}{B} \right\rfloor, \quad t = \overline{D} \bmod B, \quad t_w = \min(t, U^{\text{work}})$$

$$MRC^r = f \cdot U^{\text{work}} + t_w \tag{2}$$

In some cases this approximation can be improved even further. Recall we are using a bidirectional extension algorithm, alternating between extending paths originating from the source (forward) and the sink (backward). During the backward pass, we are essentially computing the MRC of resources from

the forward pass, and vice versa. Let us therefore iteratively extend the forward and backward paths, with D corresponding to the total number of days in the rostering period. After I^f iterations of forward extension and I^b iterations of backwards extension, we can derive the MRC of the forward resources for the final I^b days. Simply iterate over the resource vectors of partial paths stored at day $D - I^b$ and extract the maximum value for each resource.

However, the forward and backward paths are still separated by $D - I^f - I^b$ days. We therefore combine the two approximations to find the most accurate MRC per resource. This process begins by taking the maximum consumption of a resource in paths emerging from the opposite direction. We then approximate the maximum number of remaining workdays in the $D - I^f - I^b$ days separating the opposite passes using Equation (2). The sum of these values is the resource's final MRC. To reduce computational overhead, we do not compute these values for each label we extend, but only once for all labels corresponding to the same day in the roster.

4.3 Pre-dominance pruning (PDP)

Another well-known technique to accelerate dominance is to remove as many partial paths as possible in advance. Curtois et al. [4] remove equivalent paths, while Legrain et al. [12] extended this by also pruning labels that can never reach the desired objective value. They achieve this by executing a shortest reverse path search from the sink to obtain lower bounds on completions for each vertex.

Similarly, an upper bound on a resource's remaining consumption can be used to prune paths with a constraint-imposed minimum consumption. Instead of utilizing a dedicated longest path algorithm to generate upper bounds, we simply reuse the MRC from Section 4.2 to prune away paths before the dominance procedure. More formally, a path P can be discarded if for any $r \in R$ the inequality $\gamma^r(P) + MRC^r < L^r$ holds.

4.4 Dynamic bidirectional extension (DBIDI)

The bidirectional extension algorithm extends partial paths starting at both source o and sink t. This is an effective acceleration strategy as it decreases the depth of the search tree and thus significantly shrinks its overall size. One parameter to consider is the point where both passes eventually meet.

Legrain et al. [12] chose the middle of the rostering period as a natural meeting point. However, research into bidirectional extension algorithms for general SPPRCs suggests that asymmetry between the forward and backward search trees can significantly impact performance when a static midpoint is used [17]. In the SNRP, fixed days off can contribute to this asymmetry, as the number of propagated labels remains constant on those days. This in turn result in a higher number of propagated paths in either the forward or backward pass.

We therefore extend the forward or backward pass depending on which currently has fewer paths to extend, following a strategy proposed by Tilk et al. [17]. All paths are then merged once they inevitably meet.

5 Fast feasibility

Constructing an optimal single-nurse roster requires identifying the lowest-cost complete path by extending and dominating partial paths until the end of the rostering period, thereby implicitly enumerating all feasible rosters. Even with the improvements introduced in Section 4, this process can remain computationally intensive, particularly for large problem instances. Indeed, just finding a feasible solution for these instances can take considerable time. In practice, however, feasible rosters must be generated quickly.

We have identified two primary factors contributing to long computation times for large SNRP instances: (i) hard workload constraints, which make only near-complete paths feasible, and (ii) a high branching factor in the search trees due to the large number of shift types. Additional complications arise from hard constraints on consecutive days worked or off, combined with long rostering periods.

Burke et al. [4] and Legrain et al. [12] accelerate the generation of feasible rosters by limiting the number of propagated paths. The choice of which paths to propagate is crucial for the effectiveness of this beam search method. For example, Legrain et al. [12] only extend the ten paths with the lowest objective values in each iteration. We propose an alternative path selection heuristic designed for quickly producing feasible solutions. To this end, we introduce the concept of *expected workload*: for each day in the rostering period, this quantity approximates the cumulative number of hours a nurse should have worked by the end of that day in order to satisfy the workload constraints by the end of the period.

Let D denote the number of days in the rostering period and W the number of weekends. A subset of those days and weekends are not available for work due to the fixed idle days defined in the constraints. We refer to those that are available as D_a and W_a. Additionally, we denote the subset of days that are part of a weekend as D_w. Furthermore, some non-fixed weekend days will not be worked due to counter constraints on weekends. Let U^{weekend} be the maximum number of weekends a nurse can work. The average number of weekend days available in a feasible roster is then:

$$\langle D_{a,w} \rangle = \frac{D_{a,w}}{W_a} \cdot U^{\text{weekend}}$$

From this, the average number of workdays available in the rostering period is:

$$\langle D_a \rangle = D_a - D_{a,w} + \langle D_{a,w} \rangle$$

Let U^{workload} be the maximum number of minutes a nurse can work. The average load per available work day is defined by ℓ. Multiplying ℓ with the average number of available weekend days gives the total average workload on weekends ℓ_w:

$$\ell = \frac{U^{\text{workload}}}{\langle D_a \rangle}, \quad \ell_w = \ell \cdot \langle D_{a,w} \rangle$$

Since the specific weekends worked are not known in advance, this load is distributed evenly across all non-fixed weekend days. The per-day workload estimates are therefore:

$$\ell_{\text{weekday}} = \ell \text{ and } \ell_{\text{weekend}} = \frac{\ell_w}{D_{a,w}}$$

The expected workload for each day in the rostering period is computed as the cumulative sum of these daily estimates, ignoring fixed days off. During path selection, a number of paths whose cumulative workloads are close to the expected workloads are selected first to avoid paths becoming infeasible before the end of the rostering period. If this process does not result in a feasible path on a first pass from source o to sink t, we increase the number of propagated paths and perform a new extension pass.

6 Computational study

This section assesses the impact of our improved dominance rules and modified bidirectional search on the performance of the state-of-the-art algorithm introduced by Legrain et al. [12]. We enable each improvement from Section 4 individually and compare the results to those of our implementation of the original algorithm. We also evaluate the performance of the algorithm with all improvements enabled. Additionally, we measure the time it takes to produce an initial feasible solution for each instance using our expected workload path selection criterion from Section 5 while restricting the initial number of propagated paths to 64. We chose this number after experimentation and found it to be fast to compute for all different lengths of rostering periods.

For benchmarking, we used a dataset from the literature consisting of 24 instances [6]. Each instance contains as many SNRPs as there are nurses. To better reflect real-world usage of the algorithms, we solve all SNRPs in an instance 20 times, each time with a different vector of dual prices representing the value of covering a specific shift that is also part of the objective function. These dual prices were obtained by executing a column generation algorithm on each instance, and recording the dual prices in each iteration. The specific dual prices are available in a public repository[3]. The reported values correspond to the average time required to solve all SNRPs in an instance, averaged over these 20 repetitions.

Table 1 shows the reductions in computation time obtained by the improvements. The table only includes instances for which each algorithm could solve all SNRP subproblems. Negative values indicate a slowdown. The last column displays the time it took to generate a complete feasible roster with the expected workload criterion. The algorithms are implemented in the Rust programming language and experiments were run on a single core of an AMD Ryzen 9 5950X processor, with the maximum runtime set to five hours per subproblem.

[3] https://gitlab.kuleuven.be/codes/datasets/curtois2014-dual-prices

Strategy	Base	ISD	MRC	PDP	DBIDI	Combined	Feasible
Instance	Avg. time (s)	Average time reduction (%)					Time (s)
Instance 1	0.0005	-12.97	8.28	-32.97	-0.86	-9.35	0.0020
Instance 2	0.0011	0.45	23.56	8.36	13.41	26.20	0.0038
Instance 3	0.0059	14.14	23.43	9.15	17.08	29.50	0.0055
Instance 4	0.0145	6.00	49.35	31.57	5.63	53.51	0.0065
Instance 5	0.1649	-8.89	81.99	49.63	36.71	84.97	0.0106
Instance 6	0.0359	5.60	49.04	25.07	7.91	53.34	0.0164
Instance 7	0.1328	6.45	49.69	22.36	23.08	62.53	0.0159
Instance 8	16.1277	11.09	70.86	37.21	20.24	74.16	0.0228
Instance 9	0.5904	18.79	17.43	4.44	14.70	42.66	0.0333
Instance 10	3.9299	39.54	60.06	10.84	9.56	73.83	0.0367
Instance 11	0.2935	48.58	52.69	31.28	11.22	75.37	0.0565
Instance 12	1.0039	64.45	50.15	22.19	14.11	82.42	0.0997
Instance 14	1005.5752	50.50	95.91	28.40	53.96	97.95	0.1302
Instance 16	0.6222	23.94	65.29	73.15	18.70	81.78	0.0318
Instance 17	20.2740	13.93	64.90	75.72	21.78	87.57	0.0774
Instance 18	154.1834	2.39	97.03	75.58	42.51	98.34	0.0565
Instance 19	3658.5361	27.79	87.72	33.52	34.99	94.36	0.1129
Instance 20	840.3358	62.89	65.35	73.33	3.41	91.27	0.4132
Global avg.	-	**20.82**	**56.26**	**32.16**	**19.34**	**66.69**	-

Table 1. Comparison of experiments that yielded optimal solutions within 5 hours.

For each improved algorithm, the general trend is that they become more effective as the roster solution space grows. ISD peaks at a 64% improvement over the base algorithm for instance 12, which happens to be the instance with the most distinct shift types we were able to solve to optimality. Its overall improvement is similar to DBIDI at around 20%, but these methods are most effective for entirely different subproblems. Incorporating the MRC into the dominance rules provides the most effective improvement overall, with an average speedup of 56%. Combining all improvements yields an average speedup of 66% over all subproblems but exhibits a significantly larger speedup for the largest instances, with several reductions in computation time of over 90%.

Strategy	Base	ISD	MRC	PDP	DBIDI	Combined	Feasible
Instance	Number of subproblems solved						Time (s)
Instance 13	0	0	0	0	0	1	0.4040
Instance 15	10	18	20	10	20	20	0.0643
Instance 21	0	0	0	0	0	1	1.0277
Instance 22	0	0	0	0	0	0	1.4175
Instance 23	0	0	0	0	0	0	3.4389
Instance 24	0	0	0	0	0	0	33.1022

Table 2. Comparison of experiments that exceeded the 5 hours time limit.

In case one or more subproblems could not be solved by all algorithms, no relative improvement could be calculated. Instead, we compare the number of subproblems that could be solved to optimality in Table 2. These results show that our improvements can provide the additional efficiency that is needed to solve some of the larger subproblems. Indeed, either strengthening the dominance rules based on the projected MRC or merging paths at a dynamic midpoint enabled us to solve all subproblems associated with Instance 15. Combining all improvements enabled us to solve the first subproblem associated with Instance 13 and 21 to optimality. We should note that in these first subproblems, the dual prices associated with each shift are identical, enabling us to prune away many partial paths.

Although the method could not solve all subproblems to optimality, we did succeed in generating feasible solutions for all original NRP instances. This process took approximately 30 seconds for the largest instance, demonstrating the capabilities of the expected workload selection criterion with regards to generating feasible solutions.

7 Conclusion

In this paper we successfully accelerated the dominance-based path extension algorithm for SNRP introduced by Legrain et al. [12] by refining existing dominance rules and introducing new specialized pruning techniques. These improvements led to reductions in computation time of up to 98% and enabled us to solve instances larger than was possible before. Additionally, we introduced an efficient method to generate feasible rosters for the largest SNRP instances through heuristic path ordering, based on the *expected workload* of the partial paths.

Any algorithm for NRPs that relies on solving its constituent SNRPs as subproblems can benefit from our contributions, with the broader consequence of accelerating an entire family of NRP solution methods.

Acknowledgements This research was partially supported by KU Leuven, Belgium and The University of Melbourne, Australia through the KU Leuven - Melbourne Joint PhD program; by KU Leuven C24E/23/012 'Human-centred decision support based on new theory for personnel rostering' and Research Foundation — Flanders (FWO) under grant number K804824N.

References

1. Aickelin, U., Dowsland, K.A.: An indirect genetic algorithm for a nurse-scheduling problem. Computers & operations research **31**(5), 761–778 (2004)
2. Bilgin, B., De Causmaecker, P., Rossie, B., Vanden Berghe, G.: Local search neighbourhoods for dealing with a novel nurse rostering model. Annals of Operations Research **194**(1), 33–57 (2012)
3. Brucker, P., Burke, E.K., Curtois, T., Qu, R., Vanden Berghe, G.: A shift sequence based approach for nurse scheduling and a new benchmark dataset. Journal of Heuristics **16**(4), 559–573 (2010)

4. Burke, E.K., Curtois, T.: New approaches to nurse rostering benchmark instances. European Journal of Operational Research **237**(1), 71–81 (2014)
5. Ceschia, S., Dang, N., De Causmaecker, P., Haspeslagh, S., Schaerf, A.: The second international nurse rostering competition. Annals of Operations Research **274**(1), 171–186 (2019)
6. Curtois, T., Qu, R.: Computational results on new staff scheduling benchmark instances. ASAP Res. Group, School Comput. Sci., Univ. Nottingham, Nottingham, UK, Tech. Rep (2014)
7. Demassey, S., Pesant, G., Rousseau, L.M.: A cost-regular based hybrid column generation approach. Constraints **11**(4), 315–333 (2006)
8. Desaulniers, G., Desrosiers, J., loachim, I., Solomon, M.M., Soumis, F., Villeneuve, D.: A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In: Fleet management and logistics, pp. 57–93. Springer (1998)
9. Dohn, A., Mason, A.: Branch-and-price for staff rostering: An efficient implementation using generic programming and nested column generation. European Journal of Operational Research **230**(1), 157–169 (2013)
10. Gschwind, T., Irnich, S., Rothenbächer, A.K., Tilk, C.: Bidirectional labeling in column-generation algorithms for pickup-and-delivery problems. European Journal of Operational Research **266**(2), 521–530 (2018)
11. Jaumard, B., Semet, F., Vovor, T.: A generalized linear programming model for nurse scheduling. European journal of operational research **107**(1), 1–18 (1998)
12. Legrain, A., Omer, J.: A dedicated pricing algorithm to solve a large family of nurse scheduling problems with branch-and-price. INFORMS Journal on Computing **36**(4), 1108–1128 (2024)
13. Legrain, A., Omer, J., Rosat, S.: A rotation-based branch-and-price approach for the nurse scheduling problem. Mathematical Programming Computation **12**(3), 417–450 (2020)
14. Maenhout, B., Vanhoucke, M.: Branching strategies in a branch-and-price approach for a multiple objective nurse scheduling problem. Journal of scheduling **13**(1), 77–93 (2010)
15. Römer, M., Mellouli, T.: A direct milp approach based on state-expanded network flows and anticipation for multi-stage nurse rostering under uncertainty. In: Proceedings of the 11th international conference on the practice and theory of automated timetabling. pp. 549–551 (2016)
16. Strandmark, P., Qu, Y., Curtois, T.: First-order linear programming in a column generation-based heuristic approach to the nurse rostering problem. Computers & Operations Research **120**, 104945 (2020)
17. Tilk, C., Rothenbächer, A.K., Gschwind, T., Irnich, S.: Asymmetry matters: Dynamic half-way points in bidirectional labeling for solving shortest path problems with resource constraints faster. European Journal of Operational Research **261**(2), 530–539 (2017)
18. Zhang, H., Jackson, L., Tako, A., Liu, J., Dunnett, S.: Network based formulations for roster scheduling problems. In: Proceedings of the 11th International Conference on Practice and Theory of Automated Timetabling (PATAT 2016). pp. 403–419 (2016)